Solutions to SQL JOIN Problems

1. Customer Order Summary

Solution:

```
SELECT
    c.customer_id,
    c.customer_name,
    COUNT(o.order_id) AS total_orders
FROM
    customers c
LEFT JOIN
    orders o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_id, c.customer_name;
```

2. High-Spending Customers

Solution:

```
SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.order_amount) AS total_spending
FROM
    customers c
INNER JOIN
    orders o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_id, c.customer_name
HAVING
    SUM(o.order_amount) > 5000;
```

3. Orders Without Payments

Solution:

```
SELECT
    o.order_id,
    o.customer_id,
    COALESCE(p.payment_id, 'No Payment') AS payment_status
FROM
    orders o
LEFT JOIN
    payments p
ON
    o.order_id = p.order_id
WHERE
    p.payment_id IS NULL;
```

4. Product Sales by Category

Solution:

```
SELECT
    cat.category_name,
    SUM(o.order_amount) AS total_sales
FROM
    categories cat
LEFT JOIN
    products p
ON
    cat.category_id = p.category_id
LEFT JOIN
    orders o
ON
    p.product_id = o.product_id
GROUP BY
    cat.category_name;
```

5. Duplicate Orders

Solution:

```
SELECT
    o.customer_id,
    o.product_id,
    COUNT(*) AS duplicate_count
FROM
    orders o
GROUP BY
    o.customer_id, o.product_id
HAVING
    COUNT(*) > 1;
```

6. Cross-Selling Analysis

Solution:

```
SELECT
    a.customer_id,
    c.customer_name
FROM
    orders a
LEFT JOIN
    orders b
ON
    a.customer_id = b.customer_id AND a.product_id = 'Product A' AND b.product_id = 'Product B'
INNER JOIN
    customers c
ON
    a.customer_id = c.customer_id
WHERE
    b.product_id IS NULL;
```

7. Employee Sales Contribution

Solution:

```
SELECT
    e.employee_id,
    e.employee_name,
    SUM(o.order_amount) AS total_revenue
FROM
    employees e
LEFT JOIN
    orders o
ON
    e.employee_id = o.employee_id
GROUP BY
    e.employee_id, e.employee_name;
```

8. Top-Selling Products

Solution:

```
SELECT
    p.product_id,
    p.product_name,
    SUM(o.order_amount) AS total_revenue,
    SUM(o.quantity) AS total_quantity
FROM
    products p
INNER JOIN
    orders o
ON
    p.product_id = o.product_id
GROUP BY
    p.product_id, p.product_name
ORDER BY
    total_revenue DESC, total_quantity DESC
LIMIT 5;
```

9. Refund Analysis

Solution:

```
SELECT
    o.order_id,
    c.customer_name,
    COALESCE(r.refund_amount, 0) AS refunded_amount
FROM
    orders o
LEFT JOIN
    refunds r
ON
    o.order_id = r.order_id
LEFT JOIN
    customers c
ON
    o.customer_id = c.customer_id;
```